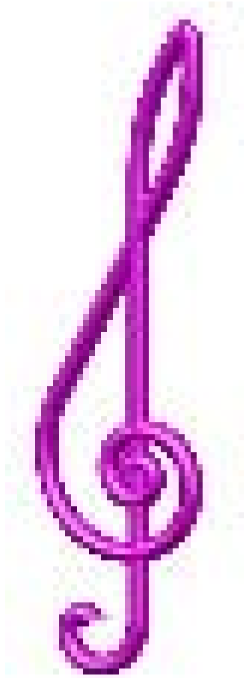
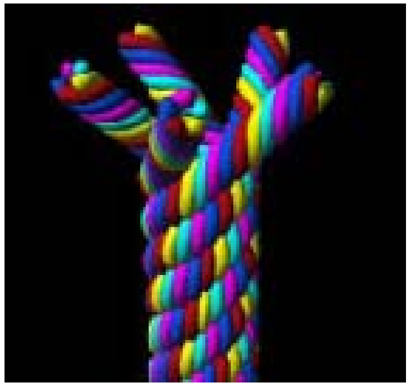
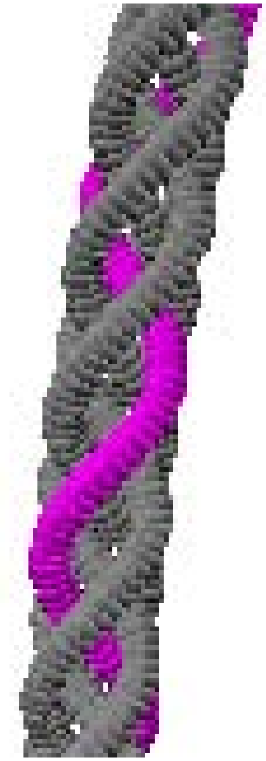




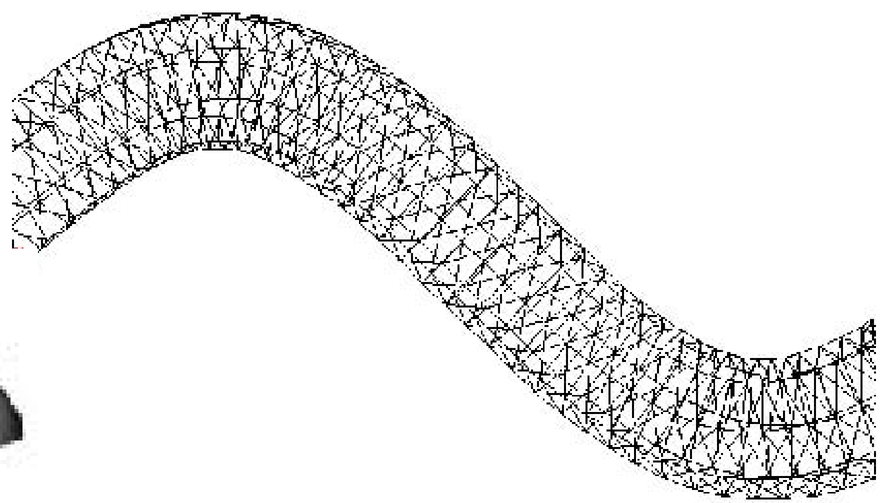


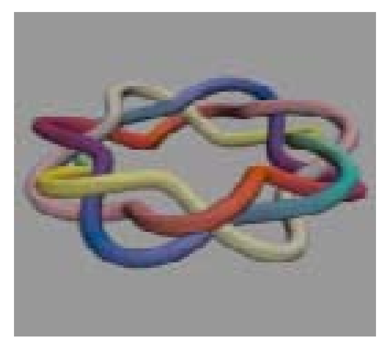
Generalized Cylinders

CSC 305

by Brian Wyvill

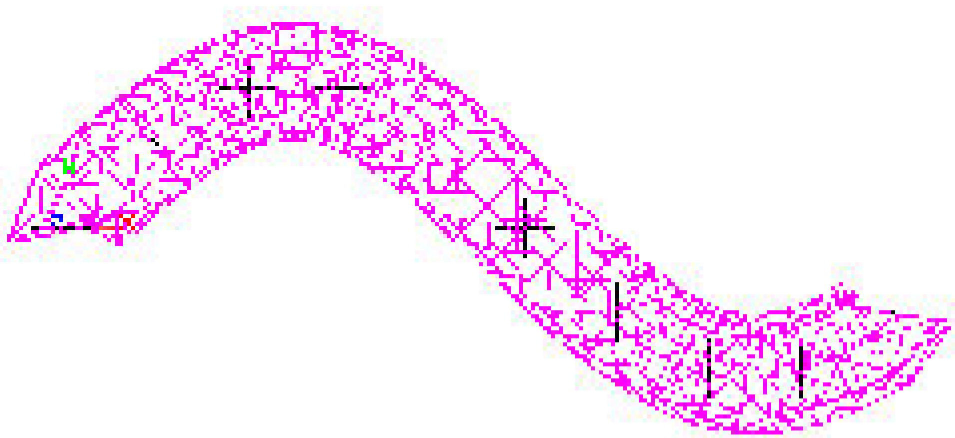
with help from Jules Bloomenthal

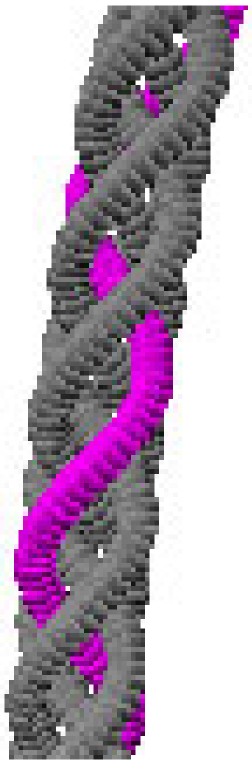




Calculation of Reference Frames Along a Space Curve
Jules Bloomenthal in Graphics Gems p567
Academic Press 1990

Also see pdf on course web page.

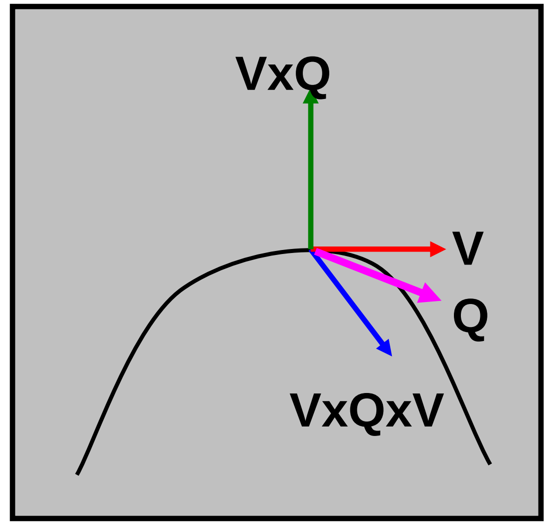




3D space curves can represent the path of an object or the boundary of a surface patch. They can also participate in various free−form constructions.

Frenet Frames

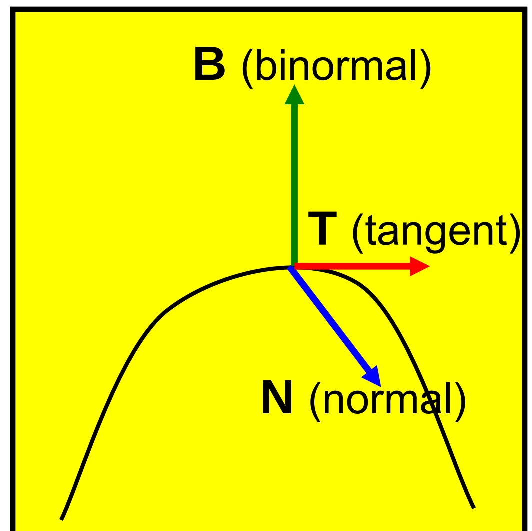
Points on a cubic
given by: $P=at^3 + bt^2 + ct + d$
Velocity: $V= 3at^2 + 2bt + c$
Acceleration: $Q=6at + 2b$



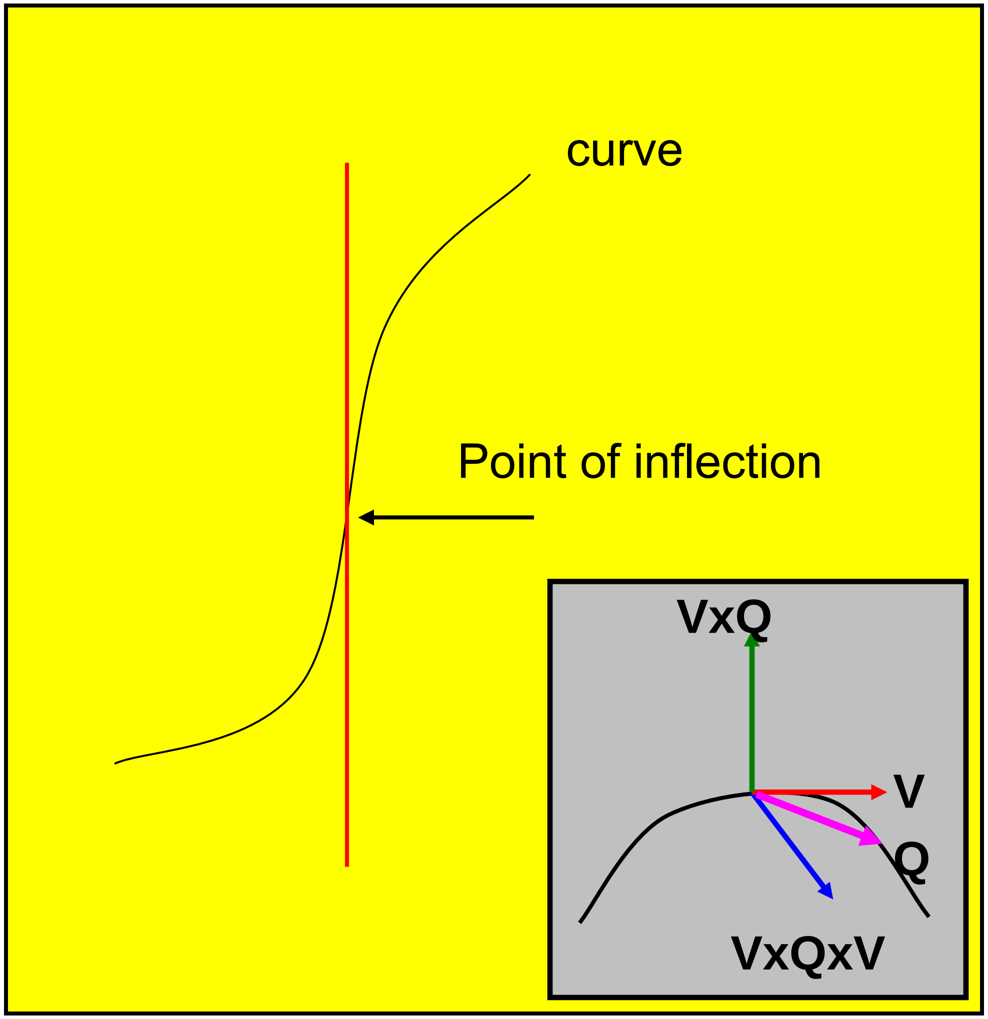
The principal normal **K** is defined in the direction of the curve as :
K = V x Q x V
strictly **(K = V x Q x V** /)**|V|**4)
(See Barsky, Beatty, Bartels)

Let **T=V/|V| N=K/|K| B=TxN**

This defines the Frenet frame. This is useful as it can be computed at arbitrary points along the curve.



Frenet Frame Problems



Consider the curve :

$f(t) = -2t^3 + 3t^2$

$f'(t) = -6t^2 + 6t$

$f''(t) = -12t + 6$

At t=0.5

$f(0.5) = -2/8 + 3/2$

$f'(0.5) = -6/4 + 3$

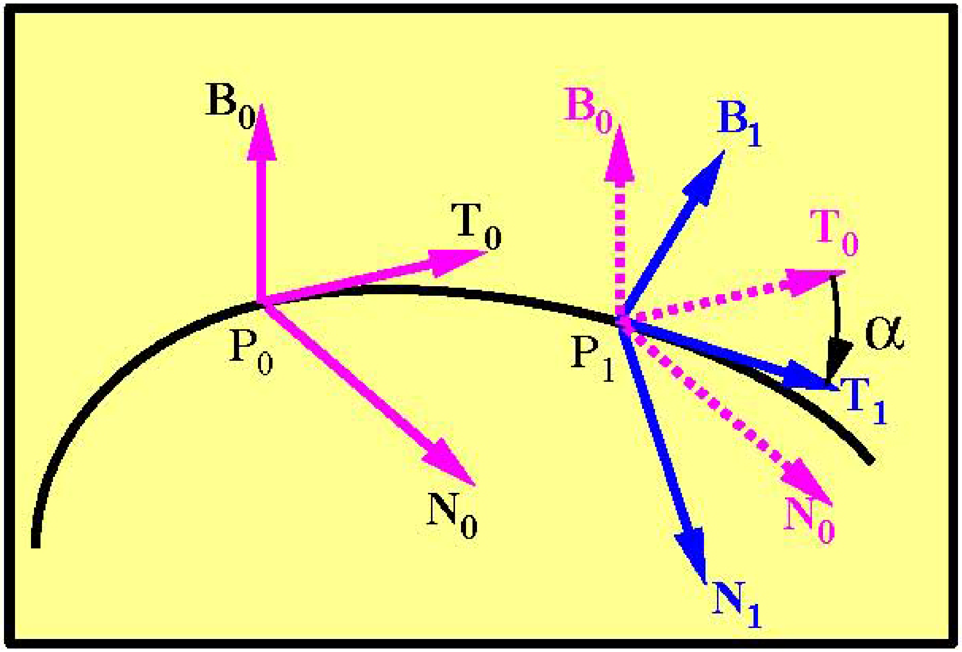
$f''(0.5) = -6 + 6 = 0$

If Q=0 we no longer have a frame.

Rotation Minimising Frames

Define an initial reference frame (with a non-zero value of f'') and compute successive frames incrementally.

This does not permit analytical computation of a reference frame

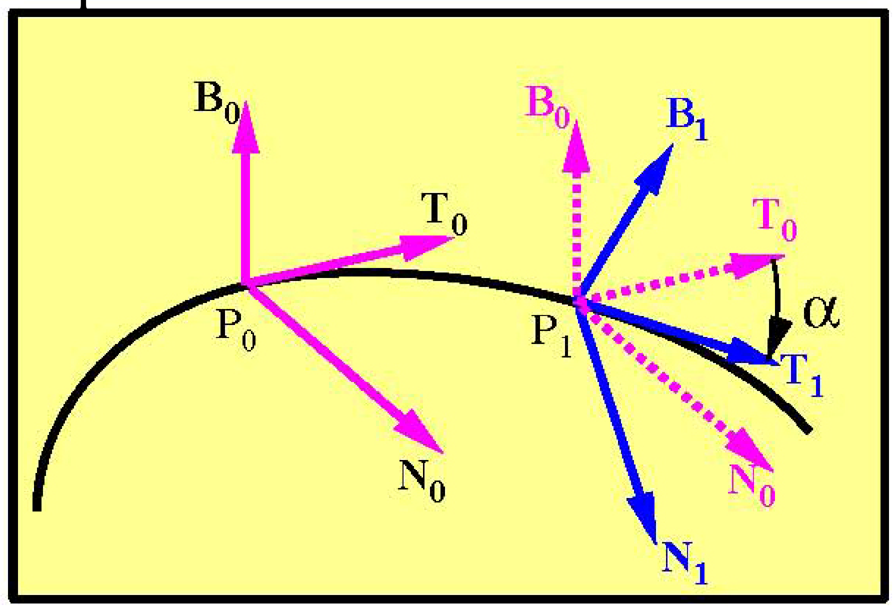


Rotation Minimising Frames

First frame: $\mathbf{f(t)} = \mathbf{a}t^3 + \mathbf{b}t^2 + \mathbf{c}t + \mathbf{d}$

Compute $\mathbf{T=V/|V|}$ $\mathbf{N=K/|K|}$ $\mathbf{B=TxN}$

If **N** is degenerate set **N** to any unit vector perpendicular to **T** then compute **B**. Subsequent frames computed from P and **T**.



Given P_i+1

$\mathbf{V}_i+1 = \mathbf{f'}(P_i+1)$

and $\mathbf{T}_i+1 = \mathbf{V}_i+1/|\mathbf{V}_i+1|$

Rotation axis: $\mathbf{T}_i \times \mathbf{T}_{i+1}$

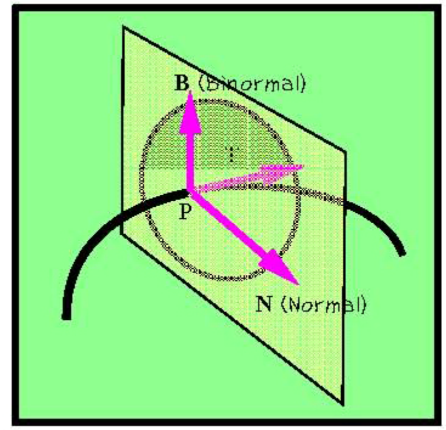
If $\mathbf{T}_i \times \mathbf{T}_{i+1} = 0$ rotation is zero.

$\mathbf{T}_i \cdot \mathbf{T}_i+1 = |\mathbf{T}_i|\,|\mathbf{T}_i+1|\cos\alpha$

$\alpha=\cos^{-1}(\mathbf{T}_i \cdot \mathbf{T}_{i+1})/(|\mathbf{T}_i|\,|\mathbf{T}_i+1|)$

$\mathbf{B}_i+1$ and $\mathbf{N}_i+1$ are computed by rotating $\mathbf{B}_i$ and $\mathbf{N}_i$

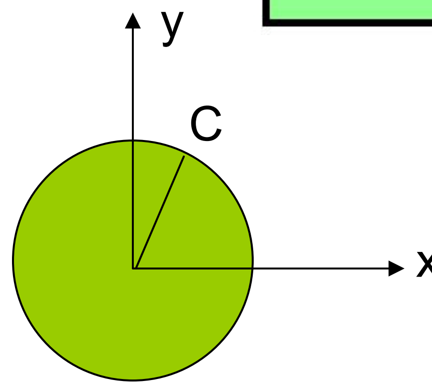
Computing the cross section



The cross section is defined on the plane formed by the normal **N** and binormal **B.**
If C(Cx , Cy) is a 2D point on the 2D cross section:
To make this 3D, on the surface of the generalised cylinder:

(Px + CxNx + CyBx Py + CxNy + CyBy Pz + CxNz + CyBz**)**
or more conveniently:

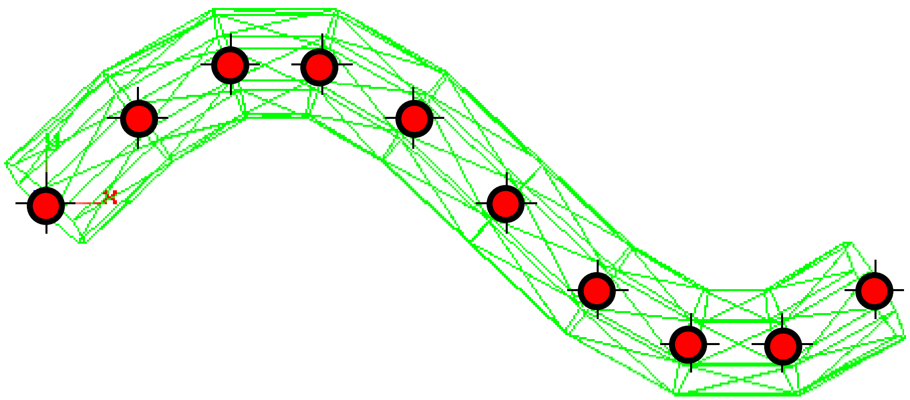
$$\begin{bmatrix} C_3x \\ C_3y \\ C_3z \end{bmatrix} = \begin{bmatrix} Nx & Bx & Px \\ Ny & By & Py \\ Nz & Bz & Pz \end{bmatrix} \begin{bmatrix} Cx \\ Cy \\ 1 \end{bmatrix}$$



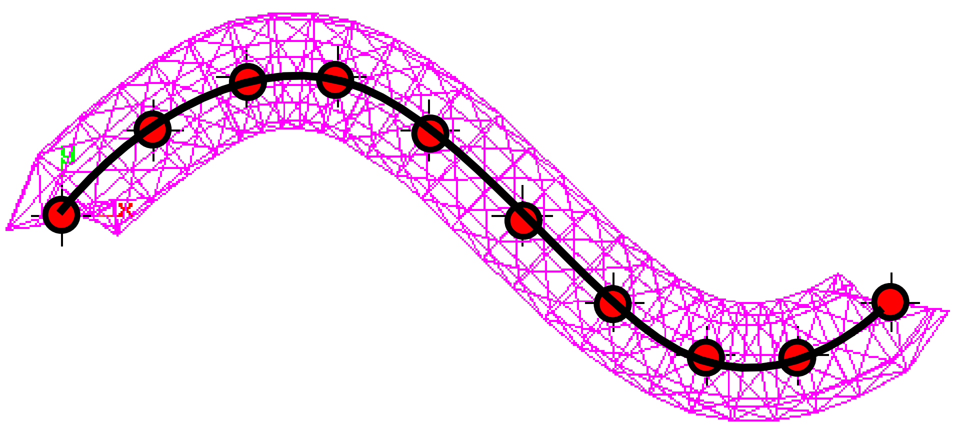
N and **B** define their own two dimensional coordinate system (i.e., they are two orthogonal axes that define a 2D plane); Cx and Cy simply state how far a point is to move in the N and B directions, respectively. That is, by multiplying Cx by **N** and Cy by **B**, you generate a point in the NB plane. The P(x,y,z) simply translates the plane to the correct point in space).

More on GC's

The results depend on the distance between successive frames.



Frame made at specified control points



Cubic interpolated between control points

Three views of cylinder with changing cross section.

Some Examples

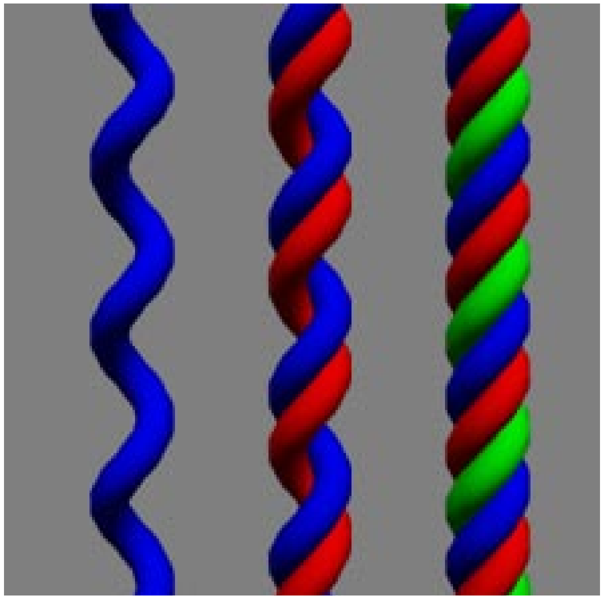
```
trebleclef 1
cyan_obj 1
cyan
0.000000 1.000000 1.000000
8
141
        0.000000 0.000000 0.000000    2 2 0
        0.000000 0.500000 0.000000    2 2 0
        0.135880 1.007111 0.000000    2 2 0
        0.411505 1.484508 0.000000    2 2 0
        0.820787 1.893790 0.000000    2 2 0
        1.347117 2.197666 0.000000    2 2 0
        1.963514 2.362829 0.000000    2 2 0
        2.633561 2.362829 0.000000    2 2 0
```

Input File format to cylinder program

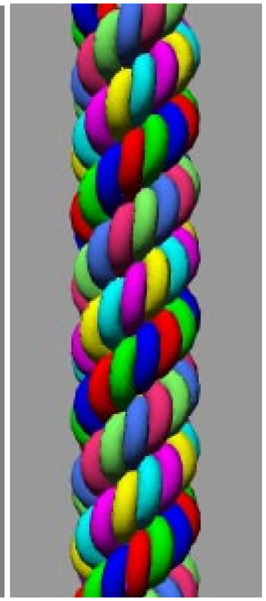
```
<pg_object_name>
        <number_of_cylinders>
        /* number_of_cylinder records containing: */
            <cylinder_name>
            <cylinder_mode (0 or 1)>
            <cylinder_color_name>
            /* if cylinder_color_name != NONE,
                                    color values follow*/
                <red_value> <green_value> <blue_value>
            <number_of_points_on_circle>
            <number_of_key_points>
            /* number_of_key_points follow containing: */
                <x> <y> <z> <scale_x> <scale_y> <twist>
                /* if cylinder_color_name == NONE,       */
                /* color for this key point is specified */
                    <keypoint_color_name>
                    <red_value> <green_value> <blue_value>
```

Hawser Laid Ropes

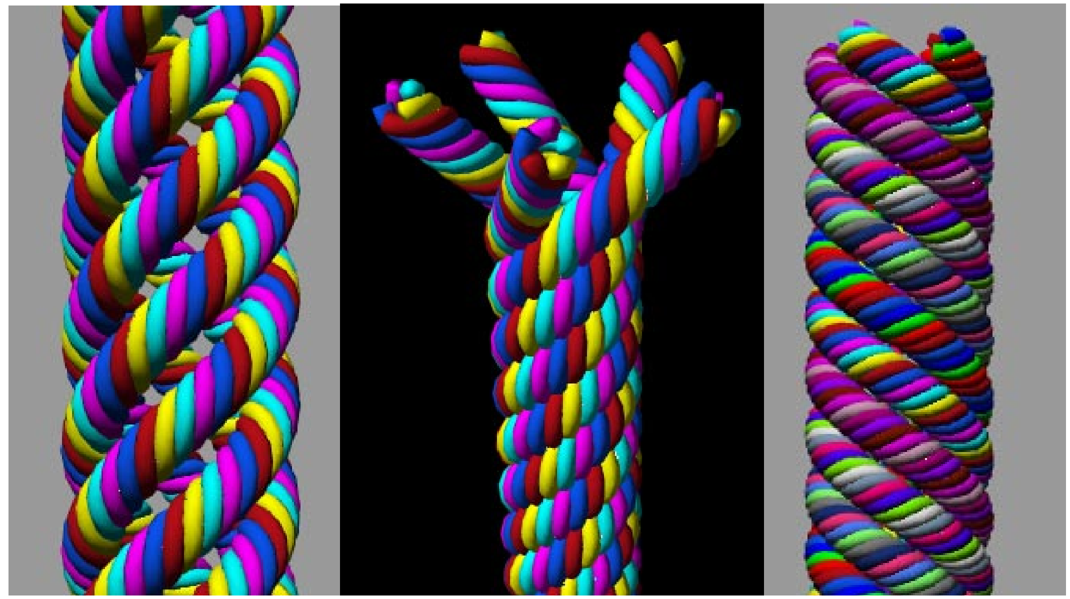
Strands follow a helix



3 strand



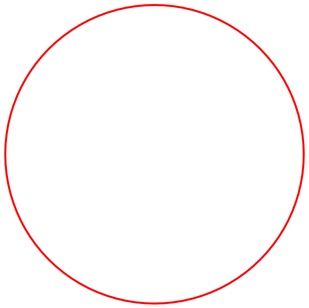
3 strands each of three strands



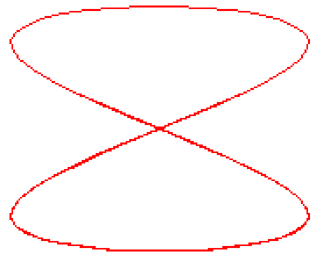
5 strands each of five strands

Braided Strands

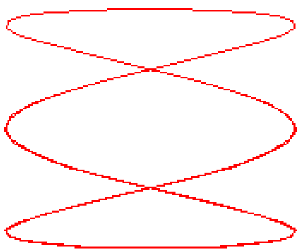
Strands follow a Lissajous Figure



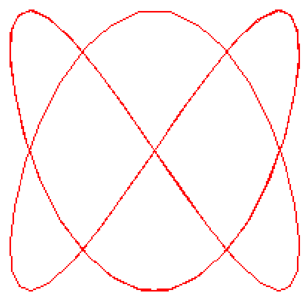
x=sin(t), y=cos(t)



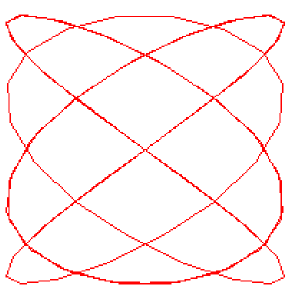
sin(2*t), cos(t)



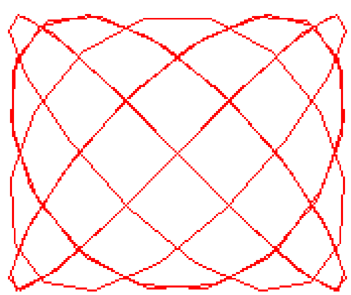
sin(3*t), cos(t)



sin(2*t), cos(3*t)

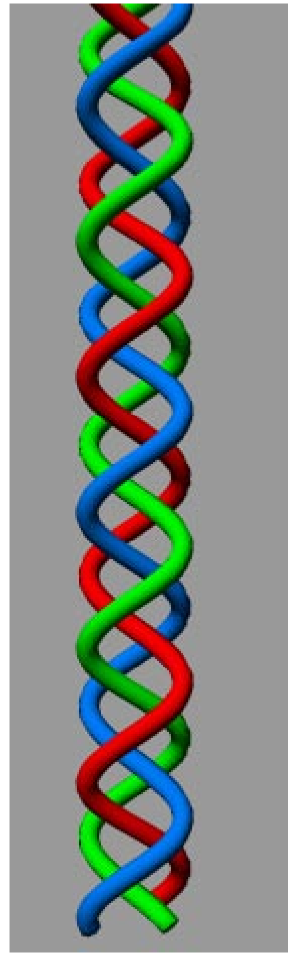


sin(4*t), cos(3*t)

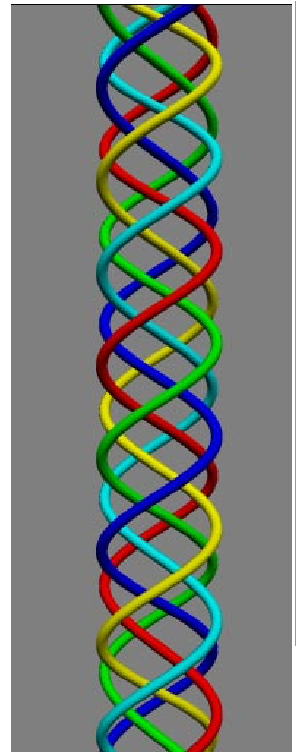


sin(4*t), cos(5*t)

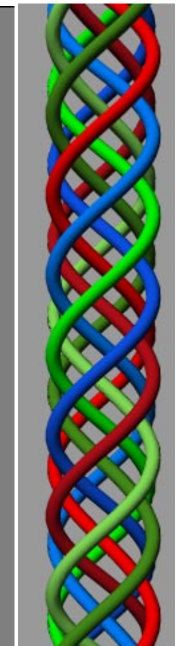
3 strands Ratio 1:2



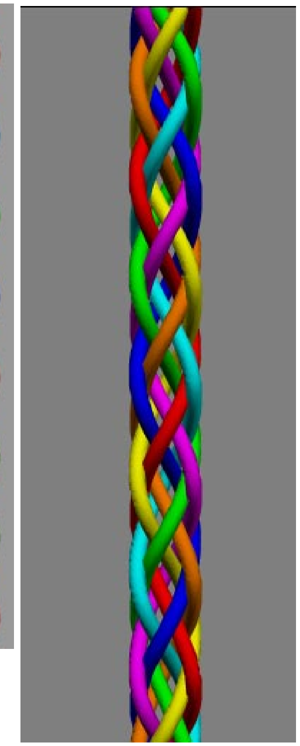
5 1:2

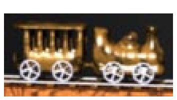


7 2:1



7 2:3

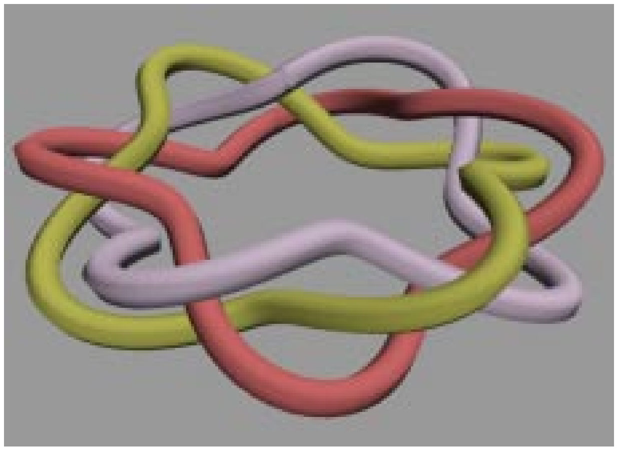




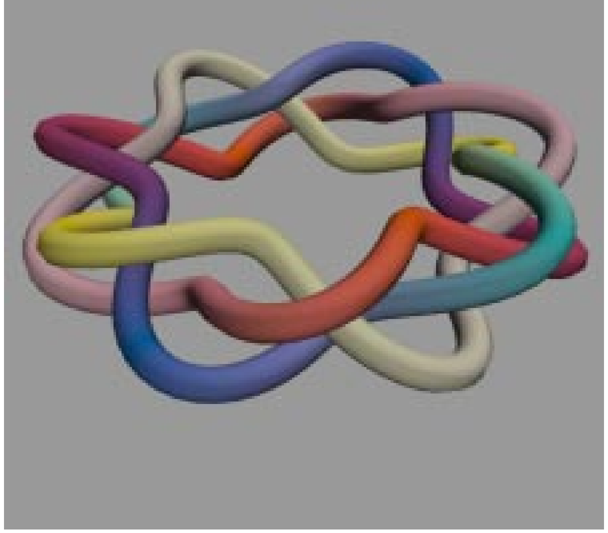
Knots

Strands follow a Lissajous Figure
twisted and bent into a circle.

3 1:2 link



3 1:2 Knot

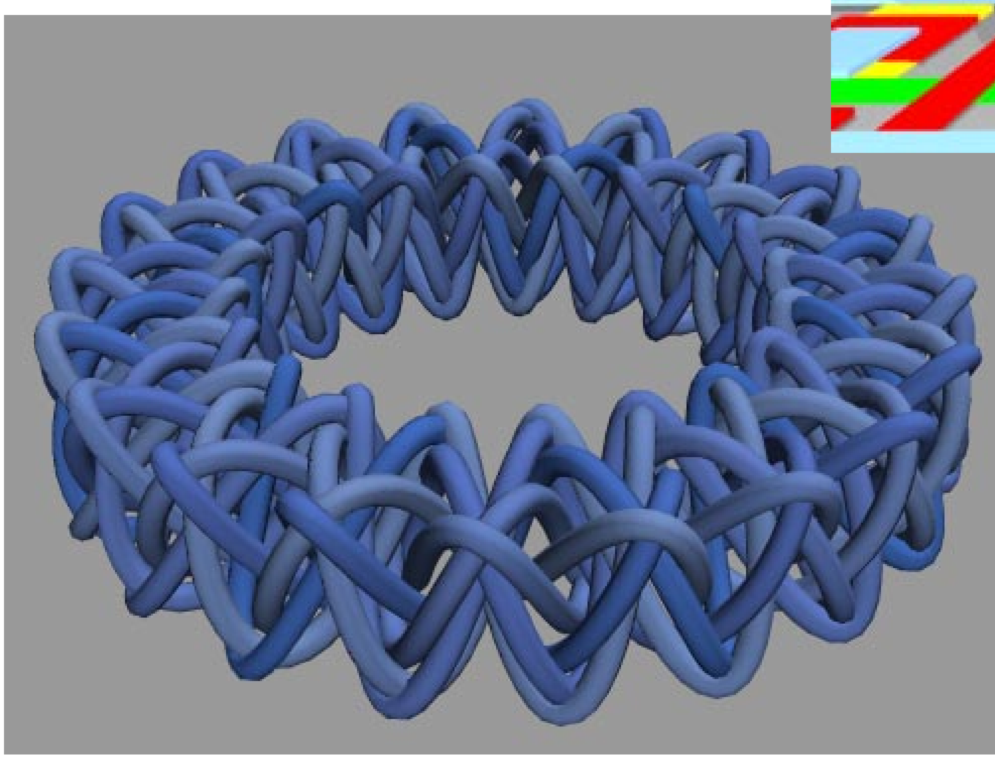


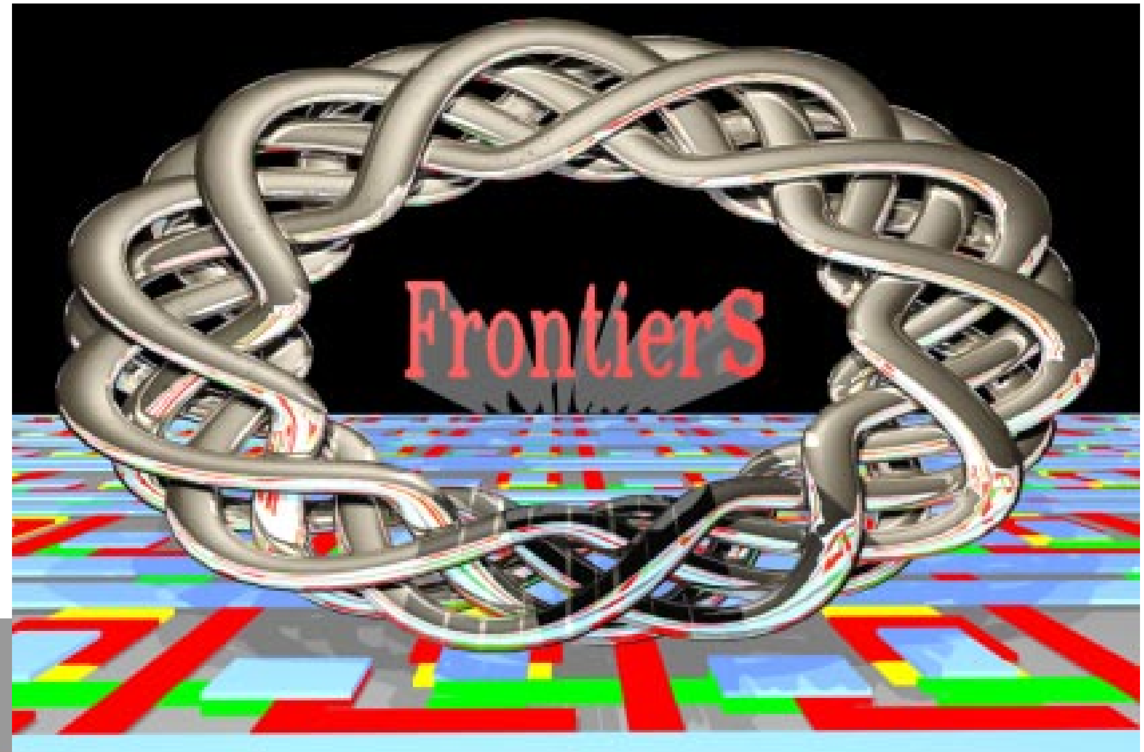
```
#define AVGERADIUS 3
rope(l, m, c, strands)  double l,m,c,strands;
{
  double interp1, interp2;
  double theta, phi,stop, cstepsize;
  double radius,x,y,z, r,g,b,cx,cy,lc;
  int i,p,q;
  i=0;   p = 0;  q = 1;
  stop = stepsize + strands*M_PI*2.0;
  printf("%d\n",1+(int) (stop / stepsize));

  cstepsize=stepsize/c;
  theta = 0.0;
  phi   = 0.0;
  while (theta <= stop) {
   theta += stepsize;
   phi   += cstepsize;
   radius = (AVGERADIUS+cos(l*phi));
   x = radius * cos(theta);
   y = sin(m*phi);
   z = radius * sin(theta);
   printf(" %10.3f %10.3f %10.3f %f %f %f\n ",x,y,z,scale_x,scale_y,twist);
   interp1 = (sin(theta/( strands)) )*(sin(theta/( strands) ));
   interp2 = 1.0 − interp1;
     cx = colour[p][0] * interp1 + colour[q][0] * interp2;
     cy = colour[p][1] * interp1 + colour[q][1] * interp2;
     lc = colour[p][2] * interp1 + colour[q][2] * interp2;
     cie_to_rgb(cx,cy,lc,&r,&g,&b);
     if (col_change) printf("knot_col%d %f %f %f\n",i++ ,r,g,b);
  }
}
```

More Examples

9 4:5 pattern





5 2:3 pattern
polygonised then ray traced